std::list

- Lists are **sequence containers** that allow **constant time insert and erase operations anywhere within the sequence**, and iteration in both directions.
- List containers are implemented as **doubly-linked lists**; Doubly linked lists can store each of the elements they contain in different and unrelated storage locations. The ordering is kept internally by the association to each element of a link to the element preceding it and a link to the element following it.
- Main drawback: lack direct access to elements by position

Singly linked list (contains 5 elements here)

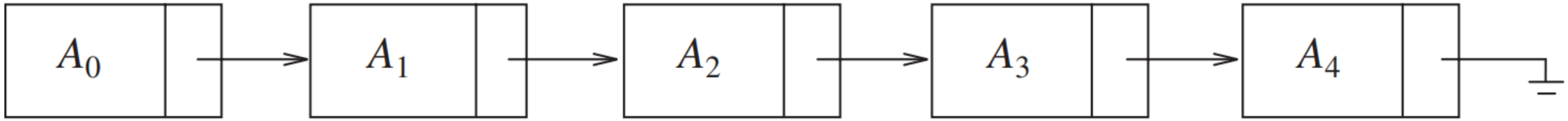


Figure 3.1 A linked list

std::forward_list in the STL

Singly linked list (deleting element A_2)

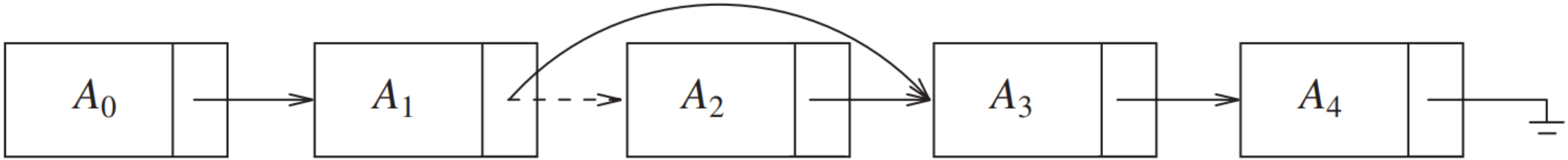


Figure 3.2 Deletion from a linked list

Singly linked list (inserting element X)

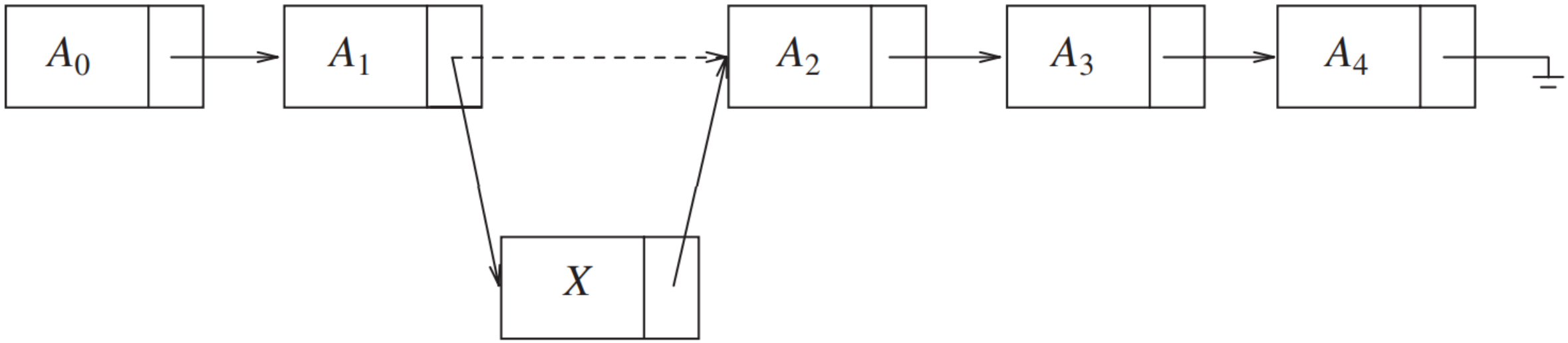


Figure 3.3 Insertion into a linked list

forward_list time complexities

- Main difference between forward_list and list: forward_list contains only a single link (to the next element) while list contains links to next and previous elements
- insert_after(position, value): $O(1)$
- pop_front(): $O(1)$
- push_front(value): $O(1)$
- remove(value): $O(n)$
- front() and end(): both $O(1)$
- Does not support pop_back
- Check rest of methods here: https://cplusplus.com/reference/forward_list/forward_list/

Doubly-linked list

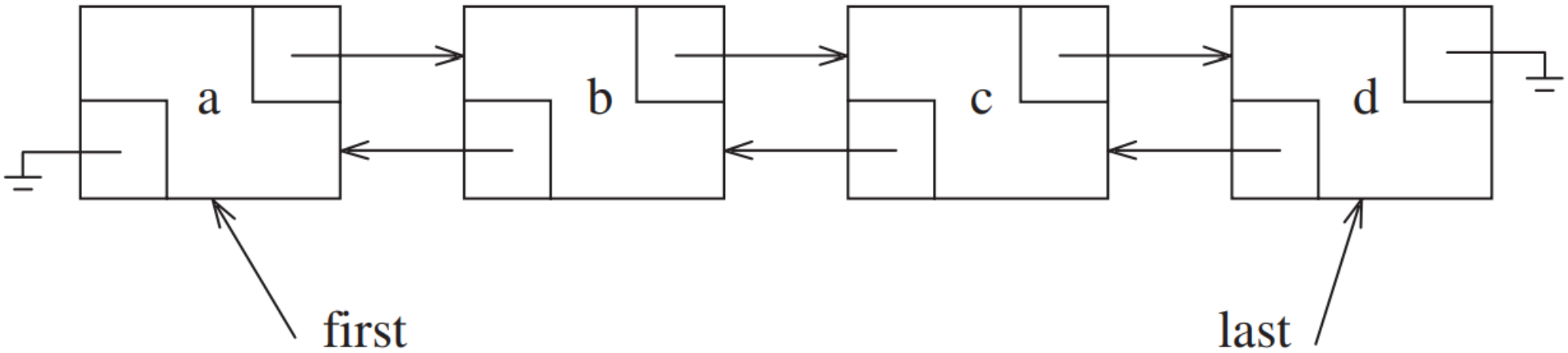


Figure 3.4 A doubly linked list

Sentinel nodes

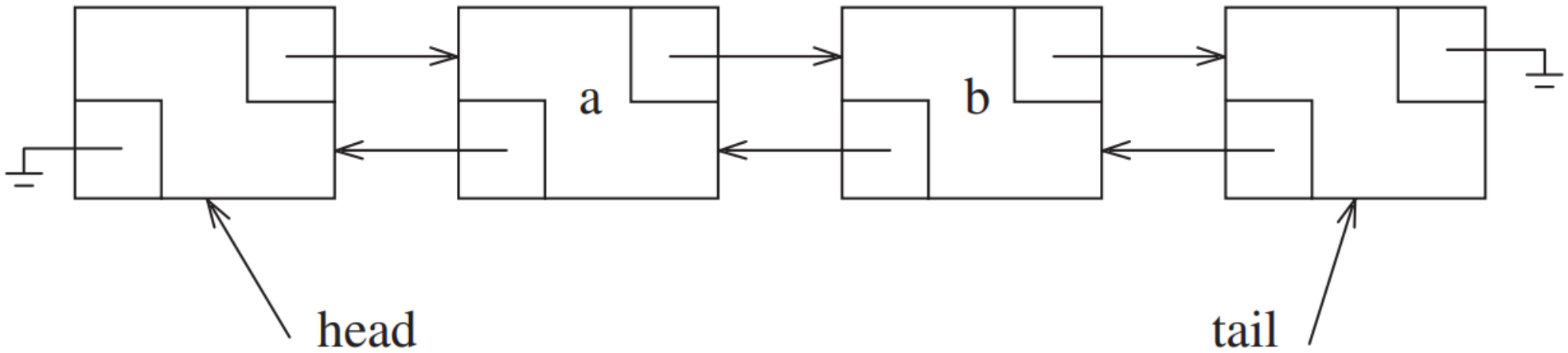


Figure 3.9 A doubly linked list with header and tail nodes

Empty doubly linked list using sentinels

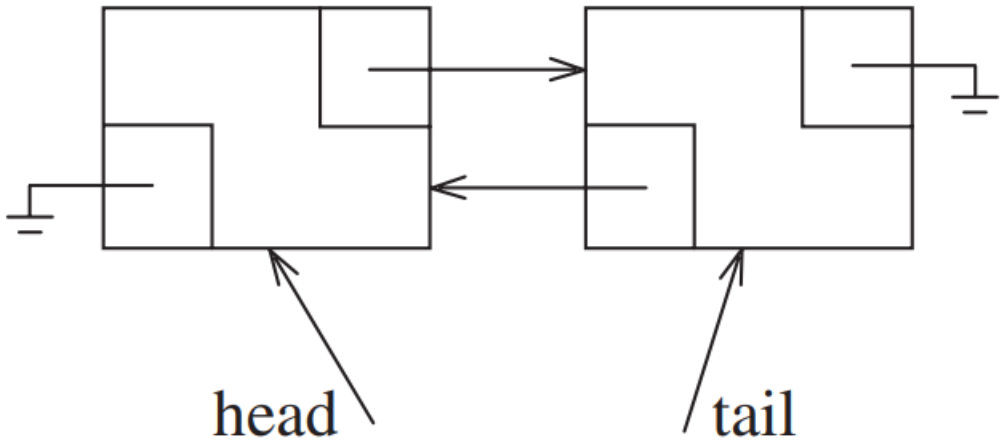


Figure 3.10 An empty doubly linked list with header and tail nodes

Inserting into doubly-linked list

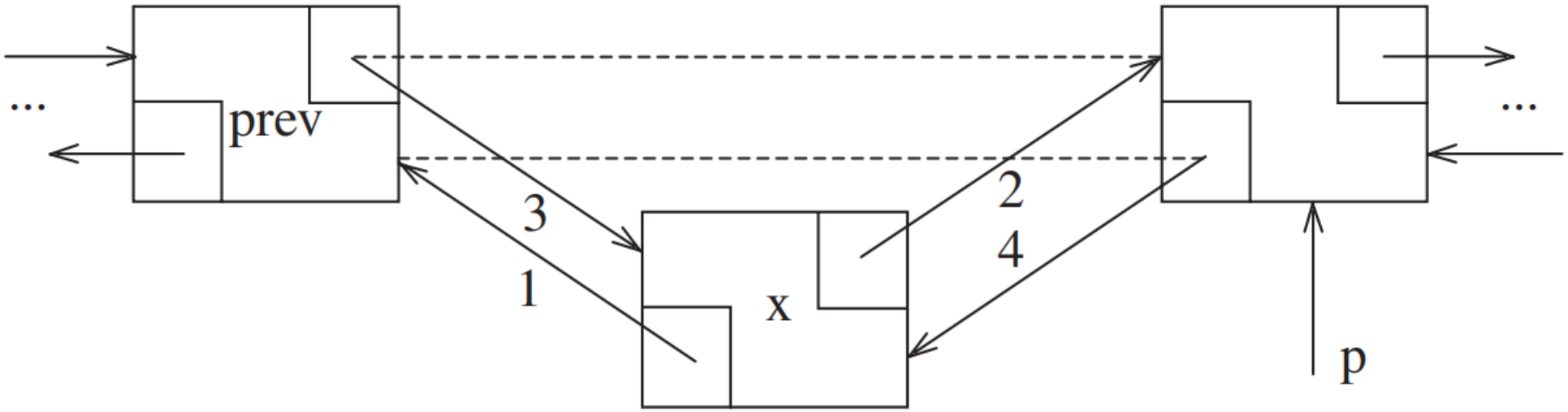


Figure 3.17 Insertion in a doubly linked list by getting a new node and then changing pointers in the order indicated

Deleting from doubly-linked list

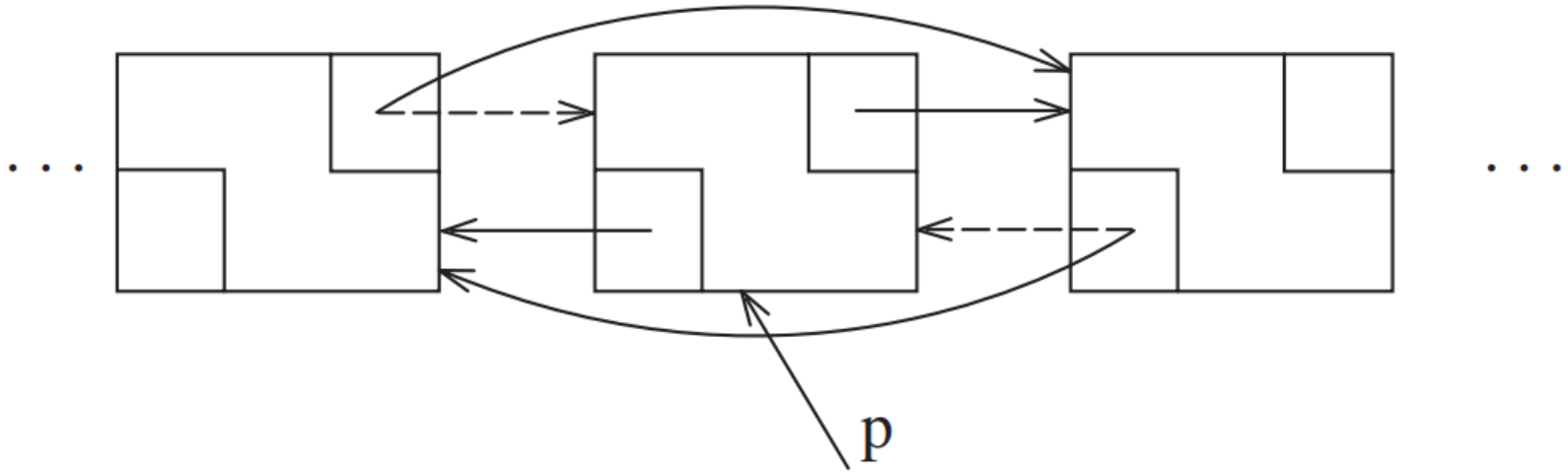
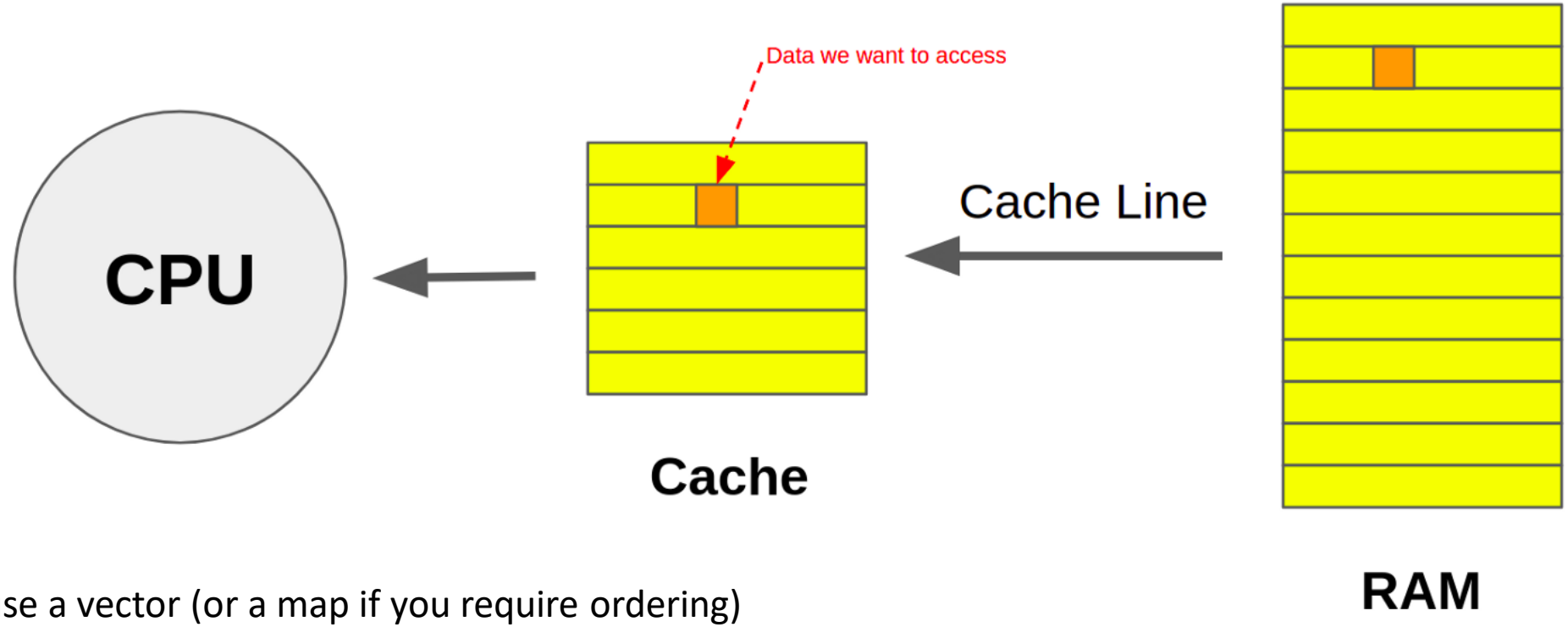


Figure 3.19 Removing node specified by p from a doubly linked list

Interesting discussion about list vs vector

- https://stackoverflow.com/questions/8742462/stdforward-list-and-stdforward-listpush-back
  - Check the top response, basically comes down to superior cache performance of vector (because it's a linear array, sequentially accessed elements are more likely to be found in the cache, vs a linked list where the elements could be anywhere in memory)



Conclusion: just use a vector (or a map if you require ordering)